Appendix A. About netCDF Data Files

A full description of the netCDF format is available from the UCAR Unidata Program Center website at http://www.unidata.ucar.edu/software/netcdf/. The following discusses what information Panoply requires in order to work with the contents of a netCDF dataset.

Panoply uses Unidata's netCDF-Java 4.0 (NJ4) code libraries* to manage its interaction with a dataset, including examination of the dataset for coordinate systems. This means that, in general, Panoply works best with netCDF datasets which have been tagged according to one of the more common defined netCDF conventions†, such as CF‡ and the older COARDS§.

Note: A dataset which follows a particular convention must explicitly indicate so by setting the dataset's global attribute named `Conventions`. If the dataset specifies no convention, then the NJ4 libraries and Panoply will default to testing whether it conforms to the relatively simple GDV convention¶.

A.1. Coordinate Variables

When examining a netCDF dataset, Panoply checks each variable to see if it has dimensions in longitude, latitude, altitude/depth, and/or time. Depending on what it finds, it will indicate whether the variable looks "plottable".

Each dimension should have an associated coordinate variable which has the same name as the dimension. For example, if the latitude dimension is called `lat` then the dataset should have a variable called `lat`. The values of this variable specify the location of the grid points along that dimension.

A.2. Boundary Variables

In addition to using the coordinate variables to identify the grid points on which the data are placed, Panoply needs to determine the area each point in the grid represents, Consequently, it also checks the netCDF dataset for associated boundary variables. This is done by testing if the coordinate variable has an attribute called `bounds` which identifies another variable containing the boundaries of the grid cells. A boundary variable will have one more dimension than its associated coordinate variable, with the second dimension having two possible values, i.e., the two boundaries.

If a coordinate variable does not have an associate boundary variable, Panoply will use values halfway between the grid points as the cell edges.

A.3. Regular Longitude and Latitude Coordinates

The usual data gridding scheme which Panoply recognizes as being plottable with longitude and latitude axes requires that the variable explicitly have dimensions of longitude and latitude. These dimensions specify a Cartesian grid. Typically the longitude dimension is named either `lon` or `longitude` and the dataset has a corresponding coordinate variable with the same name and which specifies the longitudes at the grid cell centers. Similarly, the latitude is typically named `lat` or `latitude`.

Values of the longitude coordinate variable must run eastward and be monotonically increasing. They should have units of `degrees _ east`. For example, a grid with 10° spacing and which is centered on the prime meridian would have longitude values running from -175.0 up to 175.0. Alternatively, a grid with 10° spacing which has the prime meridian as its left edge would have longitude values which run from 5.0 to 355.0. Again, note that these latitude values are at the centers of the grid cells.

---

* http://www.unidata.ucar.edu/software/netcdf-java/
† http://www.unidata.ucar.edu/software/netcdf/conventions.html
‡ http://www.cfconventions.org/
§ http://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html
¶ http://www.unidata.ucar.edu/staff/caron/Conventions.html

Values of the latitude coordinate variable may run either north to south or south to north, but they should have units of `degrees_north`. Northern hemisphere latitudes are therefore positive, and southern hemisphere latitudes negative.

For example, in CDL notation, a global data grid with 30° resolution (six latitude values and 12 longitude values) might look like:

```
dimensions:
  lat = 6;
  lon = 12;
  two = 2;
variables:
  float lat(lat);
    lat:long_name = "Latitude";
    lat:units = "degrees_north";
    lat:bounds = "lat_bounds";
  float lat_bounds(lat,two);
  float lon(lon);
    lon:long_name = "Longitude";
    lon:units = "degrees_east";
    lon:bounds = "lon_bounds";
  float lon_bounds(lon,two);
data:
  lat =
    75., 45., 15., -15., -30., -45.;
  lat_bounds =
    90., 60.,  60., 30.,  30., 0.,  0., -15.,  -15., -30.,  -60., -90.;
  lon =
    -165., -135., -105., -75., -45., -15., 15., 45., 75., 105., 135., 165.;
  lon_bounds =
    -180., -150.,  -150., -120.,  -120., -90.,  -90., -60.,  -60., -30.,  -30., 0.,
    0., 30.,  30., 60.,  60., 90.,  90.,  120.,  120., 150.,  150., 180.;
```

A.4. Projected Longitude and Latitude Coordinates

Panoply recognizes a few types of longitude-latitude data which are defined on a projected grid rather than a regular longitude-latitude grid. Such grids are typically used in regional climate and weather forecasting models and other applications which do not require global coverage.

The projected grids that are recognized and the conditions under which they may be recognized are:

- **Albers equal-area conic:** If tagged according to the CF Convention specification*.
- **Lambert conformal conic:** If tagged according to the CF Convention specification or if generated by the WRF model.
- **Mercator:** If generated by the WRF model.
- **Polar stereographic:** If tagged according to the CF Convention specification or if generated by the WRF model.
- **Rotated pole:** If tagged according to the CF Convention specification. Might be recognized in cases which do not strictly conform to the CF Convention.

A.5. Irregular Longitude and Latitude Coordinates

Another method for storing longitude and latitude data that Panoply recognizes uses two-dimensional auxiliary coordinate variables to specify the location of each and every data point. Perhaps the most common such form is a "curvilinear orthogonal" grid such as is used for some oceanographic data.

Such "irregular" data must be tagged in the netCDF file using the attributes specified by the CF Convention, i.e., the Variable to be plotted must have an attribute called `coordinates` which specifies the auxiliary coordinate variables. In the following example, shown in CDL notation, some temperature data on a 60×160 grid uses two-

* Plotting of Albers conic gridded data was added in Panoply 2.7.1. It is not available in version 2.7.

dimensional coordinate variables called `lon _ rho` and `lat _ rho` to define the longitudes and latitudes of the data points:

```
float temp(eta_rho=60, xi_rho=160);
  temp:long_name = "potential temperature";
  temp:units = "Celsius";
  temp:time = "ocean_time";
  temp:coordinates = "lat_rho lon_rho";
double lat_rho(eta_rho=60, xi_rho=160);
  lat_rho:long_name = "latitude of rho-points";
  lat_rho:units = "degree_north";
double lon_rho(eta_rho=60, xi_rho=160);
  lon_rho:long_name = "longitude of rho-points";
  lon_rho:units = "degree_east";
```

A.6. Other Longitude and Latitude Coordinate Gridding Schemes

There are other alternative longitude and latitude data storage schemes used in netCDF datasets, but it is likely that Panoply does not know how to read and plot them. Depending on whether the desired formats are described in a registered netCDF convention or are otherwise widely used, additional alternative grids can be added to Panoply. Contact the Panoply author if you would like to request that such a grid be added.

A.7. Vertical Coordinates

For latitude-vertical plots, the variable must include a dimension of regular latitude and one of the vertical. There are two ways of tagging the vertical dimension so that it may be recognized by Panoply. The easiest to recognize is when the dimension has units of pressure, and its units attribute has a string value which matches the name or abbreviation of a unit of pressure listed in the widely used UDUNITS library*. If the dimension has units of bars or millibars, the full unit name should be spelled out rather than abbreviated.

When the units of the vertical dimension are inherently ambiguous, the dimension will be treated correctly if it has an attribute called `positive`, with value of either `up` or `down`. This indicates that increasing values of the dimension unit correspond to a shift in that direction. For example, a vertical dimension with units of meters would have a `positive` attribute value of `up` if it indicates altitude, but a value of `down` if it means depth.

A.8. Time Coordinates

For time-latitude plots (keograms), the variable must have at least two dimensions, including one in time and one in regular latitude. The time axis is typically called `time`, but if not it may still be recognized if the axis's units look time-like. Panoply recognizes time dimensions denoted in "relative time" or "absolute time" units as defined in the various NetCDF conventions and should also recognize those encoded as Julian Day numbers.

A.8.1. Relative Time

A coordinate variable which has units of form *TimeUnits* `since` *TimeStamp* (for example, `hours since 1901-01-01 00:00:00`) is recognized by Panoply as being a "relative time".

A relative time variable should also include a `calendar` attribute which identifies whether a common, climate-model, or other calendar system is used. This attribute may be specified in the variable's attributes, or if not there then in the dataset's general attribute.

The recognized calendar systems are:

- **gregorian:** The standard calendar used today. However, for dates in the past it is actually a hybrid Julian/Gregorian calendar with a 10-day gap in October 1582 when much of Europe first switched from the Julian calendar to the Gregorian.

* http://www.unidata.ucar.edu/software/udunits/udunits-1/udunits.txt

- **proleptic_gregorian:** A pure Gregorian calendar, representing today's dates accurately but also extending indefinitely back into the past.
- **julian:** A pure Julian calendar.
- **noleap:** A 365-day calendar in which February always has 28 days. It might also be identified as `no_leap`, `common_year`, or `365_day`.
- **all_leap:** A 366-day calendar in which February always has 29 days. It might also be denoted `no_leap` or `366_day`.
- **360_day:** A 360-day calendar in which all 12 months have exactly 30 days each. It might also be denoted as just `360`, but such usage is deprecated.

The 365-, 366-, and 360-day fixed-length calendar systems are non-astronomical calendars which are used by various climate models. In other words, the data in such a netCDF dataset would almost certainly have been obtained by model simulation rather than through observations. The year numbering in such datasets may correspond to the year of the model run rather than to an actual calendar year.

For example, following is the CDL notation for a netCDF coordinate variable and boundary variable which describe the time information in a climate model dataset using the `noleap` calendar. These particular entries denote monthly averages for each month during one model year beginning 80 years (29,200 days) after the start of year 350 of the model run, i.e., the start of model year 430.

```
dimensions:
 time = 12;
 two = 2;
variables:
 double time(time);
  time:long_name = "time";
  time:units = "days since 0350-01-01 00:00:00";
  time:calendar = "noleap";
  time:bounds = "time_bounds";
 double time_bounds(time,two);
  time_bounds:long_name = "time interval endpoints";
data:
 time =
   29231, 29259, 29290, 29320, 29351, 29381,
   29412, 29443, 29473, 29504, 29534, 29565;
 time_bounds =
   29200, 29231,  29231, 29259,  29259, 29290,  29290, 29320,
   29320, 29351,  29351, 29381,  29381, 29412,  29412, 29443,
   29443, 29473,  29473, 29504,  29504, 29534,  29534, 29565;
```

A.8.2. Absolute Time

A coordinate variable which has units of form *TimeUnit* `as` *FormatString* (for example, `days as %Y%m%d.%f`) is recognized by Panoply as being of type absolute time.

For example, the following CDL snippet describes a time axis which denotes the twelve months of 2005:

```
dimensions:
 time = 12;
 two = 2;
variables:
  int time(time);
    time:long_name = "time";
    time:units = "day as %Y%m%d";
    time:bounds = "time_bounds";
  int time_bounds(time, two);
data:
  time =
    20050101, 20050201, 20050301, 20050401, 20050501, 20050601,
    20050701, 20050801, 20050901, 20051001, 20051101, 20051201;
```

```
  time_bounds =
    20050101, 20050131,  20050201, 20050228,  20050301, 20050331,
    20050401, 20050430,  20050501, 20050531,  20050601, 20050630,
    20050701, 20050731,  20050801, 20050831,  20050901, 20050930,
    20051001, 20051031,  20051101, 20051130,  20051201, 20051231;
```

A.8.3. Julian Day Numbers

A third unit which might be used by a time coordinate variable is the Julian Day number. Most commonly used by astronomers but sometimes employed by climatologists and meteorologists, this number is a count of the number of days that have passed since a specified epoch. A true Julian Day is the number of days which have passed since noon Universal Time on Jan. 1, 4713 BCE.

For dates in the 20th century, a true Julian Day number is large and unwieldy, so a Modified Julian Day may be used instead. This is the number of days that have elapsed since midnight Universal Time at the start of Nov. 17, 1858. The formula for converting between a Julian Day, `JD`, and a Modified Julian Day, `MJD`, is:

```
MJD = JD – 2,400,000.5
```

For example, the CDL text for a time dimension using Modified Julian Days for the units and denoting 12:00 a.m. on the dates starting the months of 2001 might look like:

```
dimensions:
  time = 12;
variables:
  int time(time=12);
    time:units = "Modified Julian Day";
data:
  time =
    51910, 51941, 51969, 52000, 52030, 52061,
    52091, 52122, 52153, 52183, 52214, 52244;
```

Although the above example uses integral values, the Julian Day can be a fractional number.

A.11. Special Values

If a variable has invalid or missing values then the appropriate attributes `valid _ min`, `valid _ max`, `valid _ range`, and/or `missing _ value` should be specified so that Panoply treats them correctly.

Panoply treats all missing or invalid values the same and, when rendering a plot, draws them all using a single invalids color (typically a shade of gray) as selected in the plot controls.

A.12. Scale Labels

Panoply uses the variable's `long _ name` and `units` attributes, if they are provided, for the default labeling of the plot scale. If the `long _ name` is missing, Panoply checks to see if a `standard _ name` has been provided instead.